Intrusion Detection System Using Genetic Algorithm

Ch.Satya Keerthi.N.V.L[#1], P.Lakshmi prasanna[#2], B.Minny Priscilla[*3], M.V.B.T.Santhi[#4]

[#]*IST Department, K.L.University*
Greenfields, Vaddeswaram, Pincode: 522502, A.P, India
[*]*M.TECH(CSE),Student, K.L.University*
Greenfields, Vaddeswaram, Pincode: 522502, A.P, India

***Abstract*—This paper shows how network connection information can be modeled as chromosomes. The objective of the system is to create a new set of rules during run time. So the intruder cannot be able to attack the system with virus. The Intrusion Detection System in Networking Using Genetic Algorithm (IDS) is to identify the intruder and block the data from the intruder to avoid the system attack by the virus. This new system is a replacement of the existing system. In existing system, at run time it will not create a set of rules. The major components of the system are creating new set of rules during run time.**

***Keywords*: Intrusion detection, Genetic algorithm, Data set, Client, Server.**

I. INTRODUCTION

The Intrusion Detection System in Networking Using Genetic Algorithm (IDS) is for Global Techno Solutions. The main objective of this system shows how real time network connection behavior can be modeled as chromosomes and how the parameters in genetic algorithm can be defined in this respect. This document is intended for the various user groups like Network Administrators, Developers, Project Managers, Marketing staff and user.

In existing system, at run time it will not create a set of rules. The objective of the system is to create a new set of rules during run time. So the Intruder cannot be able to attack the system with virus. Unlike other implementations of the same problem, this implementation considers both temporal and spatial information of network connections in encoding the network connection information into rules in IDS. This is helpful for identification of complex anomalous behaviors. This work is focused on the TCP/IP network protocols.
Computer networks may be vulnerable to many threats along many avenues of attack, including:

- Social engineering, wherein someone tries to gain access through social means pretending to be a legitimate system user or administrator, tricking people into revealing secrets, etc.)
- War dialing, wherein someone uses computer software and a modem to search for desktop computers equipped with modems that answers, providing a potential path into a corporate network
- Denial-of-service attacks, including all types of attacks intended to overwhelm a computer or a network in such a way that legitimate users of the computer or network cannot use it
- Protocol-based attacks, which take advantage of known (or unknown) weaknesses in network services
- Host attacks, which attack vulnerabilities in particular computer operating systems or in how the system is set up and administered
- Password guessing

Internet firewalls have been around for a hundred years—in Internet time. Firewalls can help protect against some of these attacks, but certainly not all. Firewalls can be very effective at what they do. The people who set up and use them must have the knowledge of how they work, and also be aware of what they can and cannot protect. An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

GA is used to generate artificial intelligence rules for IDS. The IDS can be viewed as a rule-based system (RBS) and GA can be viewed as a tool to help generate knowledge for the RBS. In recent years, Intrusion Detection System (IDS) has become one of the hottest research areas in Computer Security. It is an important detection technology and is used as a counter measure to preserve data integrity and system availability during an intrusion. An Intrusion Detection System is a system for detecting intrusions and reporting them accurately to the proper authority. Intrusion Detection Systems are usually specific to the operating system that they operate in and are an important tool in the overall implementation an organization's information security policy, this reflects an organization's statement by defining the rules and practices to provide security. A methodology of applying genetic algorithm into network intrusion detection technique is unique as it considers both temporal and spatial information of network connections during the encoding of the problem; therefore, it should be more helpful for identification of network anomalous behaviors.

II. RELATED WORK

A. Existing System

- Traditional systems in place for intrusion detection primarily use a method known as "fingerprinting" to identify malicious users. They are complex.
- They are rules dependent .If the behavior of the packets flowing in the network is new, then the system cannot take any decision. So they purely work in the basis of the initial rules provided.
- It cannot create its own rule depending on the current situation.
- It requires manual energy to monitor the Inflowing packets and analyze their behavior.
- It cannot take decisions in runtime.

B. Proposed System

- It uses Genetic algorithm, which an artificial intelligence problem-solving is based on the theory of Darwinian evaluation applied to mathematical models.
- IDS compare learned user characteristics from an empirical model to all users of a system.
- It includes both temporal and spatial information of the network traffic in the rule set.
- It is both network based and host based system.
- It can take decisions in runtime.

C. Advantages

- It eliminates the need for an attack to be previously known to be detected because malicious behavior is different from normal behavior by nature.
- Using a generalized behavioral model is theoretically more accurate, efficient and easier to maintain than a fingerprinting system.
- It uses a constant amount of computer resources per user, drastically reducing the possibility of depleting available resources.
- Once installed, there is no need for any manual energy to monitor the system.
- It generates its own rules depending on the real-time behavior of the packet.
- It dynamically increases the rules in the dataset according to the packets flowing in the network and the decisions taken by the system. Due to the increase of rules in the rule set, the reliability of the system also increases.

III. SYSTEM DESIGN

A. Overview of Genetic Algorithm

Genetic Algorithms are a searching algorithm designed to mimic the way nature reproduces itself and betters itself in doing so. Genetic algorithms have several individuals in its population, each one of which could be a potential solution to a problem. Each one of those individuals would be following a path to a possible solution, which means that it is even possible for the search to find more than one solution. Different researchers have produced many Genetic Algorithms, and all of them are very different from each other. They all, however, display the characteristics of the genetic algorithm, which follow these basic steps:

1. Step one it to randomly create a population of individuals.
2. Step two is to evaluate the population to see which individuals will contribute to the next generation.
3. Step three is to alter the new generation of individuals once they have been paired off.
4. The final step is to discard the old population and perform step two on the new population.

Once step three, above, has been completed, the algorithm jumps back to step two. The loop will only stop when one of the individuals has been evaluated and is said to be either very close to the solution, or it has found the solution.

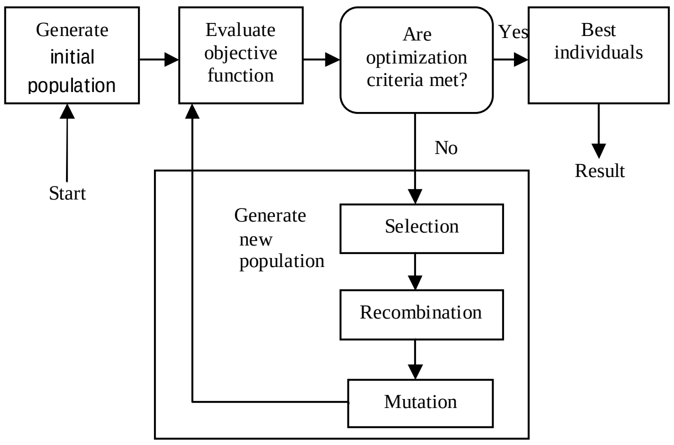

Fig1: Genetic algorithm

Once the new population has been created, it is time to alter the individuals so that they are different from the other generation. There are two possible operations, which can be performed here, crossover and mutation.

1) Crossover

This occurs when two individuals, which have been paired off, exchange chromosomes. A random number is generated. The number must be between 1 and the 1-(the maximum number of bits in the bit string). The bit string is considered to be the individual and the bits are the chromosomes. Once the random number has been generated, then all digits after that position in the bit string are exchanged.

2) Mutation

Another change an individual may go through is known as mutation. Unlike crossover, it only involves one individual. Depending one what algorithm it is (there are many genetic algorithms), mutation is not very likely to occur. Usually, the possibility of a mutation occurring to one of the chromosomes is set to 1 in thousands.

There are many different genetic algorithms, some of which do not incorporate mutation, and when they do, they handle mutation in a different way to the rest! The basic example may have the computer generate a random number, which decides whether or not a bit is to undergo mutation. Once a bit is selected for mutation, some algorithms simply flip the bit. That is, if the chromosome is a 1, then the algorithm will change it to a 0, and visa versa.

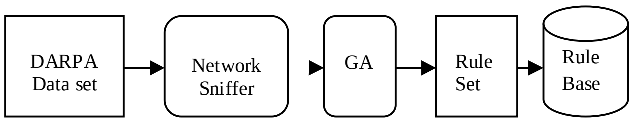

Fig2: System architecture

The Figure shows the structure of this implementation. We need to collect enough historical data that includes both normal and anomalous network connections. This is the first part inside the system architecture. The network sniffers analyze this data set and results are fed into GA for fitness evaluation. Then the GA is executed and the rule set is generated. These rules are stored in a database to be used by the IDS.

3) Rule Base

It is a Database of rules whenever the user tries to malpractices the system will generate dynamic rules set taken from the rule Database such as port number of destination, The purpose of transaction, association with the destination, number of times visited etc.

4) Creating rules in Dataset

The Rules (Both Normal and Anomal) are created in the dataset, as the chromosomes for matching with the real time connection. The administrator can just specify the attributes that he thinks to be both normal and abnormal in the specified screen provided. The Entered behaviors are automatically converted into Chromosomes, and stored in the rule set to which he specified. These rules are responsible for providing knowledge for both the Ids and Genetic algorithm.

5) Three Rounds of Generation

There are three rounds of RULE generation in this paper. This function which produces more and more new rules and add them to the existing rules. The main functional advantage here is that every time this generation of rules takes place the number of rules overall simply doubles and also the administrator need not keep account of all these rules.

The major modules in this paper include the following:

a) Client: This module is responsible for the client side communication system interface. This module has the client program and the hop count program referred by in it.
b) Server: This is the server side interface which is preset in the server system and is solely under the control of the administrator. Any transaction in the network will be monitored by the Server.
c) Hop – count: This module deals with the routing of the internal message within the network. It specifies the intermediate systems.
d) Passer: this module deal with handling the messages received from and external network and in is routing to the mentioned system.

IV. Verification and Validation

A. Component Test Plans, Procedures and strategy overview

It contains five specific points that will be used to describe each component of the "Intrusion Detection System in Networking Using Genetic Algorithm" paper.

Component testing is performed after completion of each component. The most-recently completed component is tested, and the other already-completed components are re-tested to verify that integration of the new component did not introduce any defects. A component will not be considered completed until it has successfully passed all component tests and is verified to have fulfilled the requirements set in the Software Requirements Specification document.

The methods that we will be using during the testing:

- Verify the results according to the input.
- Verify the transitions between the pages.
- Make sure the settings and all options works properly.

TABLE I. Component Client

Test case group identification	Client
Functions to be tested	Functions Tested Include • Buttons to Select o Send o Clear o Hopcount
Testing approach	Testing whether the buttons are navigating to the correct pages and producing the proper results.
Pass/Fail criteria	The buttons should navigate to the correct pages and should produce the correct results.
Testing Purpose	On click of buttons check whether it is directed to valid references.
Individual test cases	**Test case 1:** **Test case identifier**: Hopcount Button • **Input –1:** User enters a valid Intermediate System number between 0 to 5 for the process of IP address to start its working. • **Expected output-1**: Proceed. • **Input-2**: User enters an invalid Intermediate System Number to start its working. • **Expected output-2:** Display an error message and ask for reentry. • **Environment:** Java, Windows Platform. • **Precedence and dependencies:** This test case has to perform at first itself. This test case has no dependencies. **Test case 2:** **Test Case Identifier**: Send Button

	• **Input –1:** User enters the valid IP address, port number and Message. • **Expected output-1:** Proceed. • **Input –2:** User enters the invalid IP address, port number. • **Expected output-2:** Display an error message and ask for reentry. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test case has to perform at first itself. This test case has no dependencies. **Test case 3:** **Test Case Identifier**: Clear Button • **Input:** User selects the specified button. • **Expected output** o The link shows to clear the available data from the text fields. • **Environment** o Java, Windows Platform • **Precedence and dependencies** o This test case has to perform at first itself. This test case has no dependencies.

TABLE II. COMPONENT PASSER

Test case group identification	Passer
Functions to be tested	Functions Tested Include the main functions for o Receive o Check o Send
Testing approach	Testing whether the IP address, port number and message are restricted user or not.
Pass/Fail criteria	The passer has to process the IP address, if it is not restricted user.
Individual test cases	• **Test case identifier**: passer • **Input-1:** Receive the valid IP address, Port number and message. • **Expected output-1** o Proceed • **Input-2:** Receive the invalid IP address, Port number and message. • **Expected output-2** o It will display that it is a restricted users. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test can be done after the client process.

TABLE III. COMPONENT HOP COUNT

Test case group identification	Hopcount
Functions to be tested	Functions Tested Include • Buttons to Select o Store o Process o Back
Testing approach	Testing whether the buttons are navigating to the correct process and proceeding to send the data.
Pass/Fail criteria	The buttons should navigate to get the correct IP address.
Individual test cases	**Test case 1:** **Test case identifier**: Store Button • **Input-1:** User enters the valid intermediate system number and names. • **Expected output –1:** o Proceed. • **Input-2:** User enters the invalid intermediate system names greater than five. • **Expected output-2:** o Display the error message and ask to reentry. • **Environment** o Java, Windows Platform. • **Special procedures** o Check whether all the fields have System names, if not show an error message and retain the control in same page itself. • **Precedence and dependencies** o This test case has to perform at first itself. This test case has no dependencies. **Test case 2:**

	Test Case Identifier: Process Button • **Input:** User selects the specified button. • **Expected output** o The link shows the process of the intermediate system names. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test case has to perform at first itself. This test case has no dependencies. **Test case 3:** **Test Case Identifier**: Back Button • **Input:** User selects the specified button. • **Expected output** o The link shows the process back to the client page. • **Environment:** Java, Windows Platform. • **Precedence and dependencies:** This test case has to perform at first itself. This test case has no dependencies.

TABLE IV. COMPONENT IDS

Test case Group Identification	IDS
Functions to be tested	Functions Tested Include the main functions for o receive o getHeaderInformation o hopcheck o chromoConvert o checkExactAnamolMatch o checkExactNormalMatch o geneticAlgorithm o finalDecision
Testing approach	Testing whether the user is restricted user or not.
Pass/Fail criteria	The IDS has to process the incoming of the real time behavior and convert into the chromosomes and send the message to the particular destination.
Individual test cases	• **Test case identifier**: ids • **Input:** Receive the IP address, Port number. • **Expected output** o The link shows the process of the genetic algorithm. • **Environment** o Java, Windows Platform. • **Precedence and dependencies:** This test can be done after the passer process.

TABLE V. COMPONENT CHROMO CONVERT

Test case group identification	Chromoconvert
Functions to be tested	Functions Tested Include the main functions for o receive o convert
Testing approach	Testing whether the chromosomes are matched or not and navigating to the correct match and producing the proper results.
Pass/Fail criteria	The IDS has to process the incoming of the real time behavior and convert into the chromosomes.
Testing Purpose	Testing whether the chromosomes are matched or not and navigating to the correct match and producing the proper results
Individual test cases	• **Test case identifier**: chromoConvert • **Input:** Receive the IP address, Port number. • **Expected output** o The source IP address, destination IP address and port number are converted into chromosome and stored in the dataset. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test can be done after the client process.

TABLE VI. COMPONENT GENETIC ALGORITHM

Test case group identification	Genetic Algorithm
Functions to be tested	Functions Tested Include the main functions for o recieveRealChromo o anamolDataset o anamolGenetic o normalDataset o normalGenetic o decision
Testing approach	Testing whether the converted chromosomes are matched or not. By using genetic algorithm, during run time new rules are created and added into the dataset and producing the proper results.
Testing Purpose	Testing whether the chromosomes are matched. If it is not matched, then the new rules are generated.
Pass/Fail criteria	orithm has to process the new rules are generated

Individual test cases	**Test case 1:** • **Test case identifier**: genetic Algorithm • **Input:** receive IP address and port number. • **Expected output** o The link will take directly to the anamol dataset or normal dataset according to the real time network behavior. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test can be done after the passer process.

TABLE VII. COMPONENT NORMAL DATASET

Test case group identification	Normal Dataset
Functions to be tested	Functions Tested Include the main functions for o readNormalFile o convertChromo
Testing approach	Testing whether the chromosomes are matched in the normal file and convert into chromosomes.
Pass/Fail criteria	The normal dataset has to process the incoming of the real time behavior and convert into the chromosomes by fitness and recombination.
Individual test cases	• **Test case identifier**: Normaldataset • **Input:** receive converted chromosomes. • **Expected output** o The new dataset rules are created and added in the Normal dataset. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test can be done after the chromoconvert process.

TABLE VIII. COMPONENT ANAMOL DATASET

Test case group identification	Anamol Dataset
Functions to be tested	Functions Tested Include the main functions for o readAnamolFile o convertChromo
Testing approach	Testing whether the chromosomes are matched or not in the anamol dataset, else it will create the new data rules and navigating to the correct match and producing the proper results.
Pass/Fail criteria	The anamol dataset has to process the incoming of the real time behavior and convert into the chromosomes by fitness and recombination.
Individual test cases	• **Test case identifier**: Anomaldataset • **Input:** receive the converted chromosomes. • **Expected output** o The new dataset rules are created and added in the anamol dataset. • **Environment** o Java, Windows Platform. • **Precedence and dependencies** o This test can be done after the chromoconvert process.

V. IMPLEMENTATION

Implementation is most crucial stage in achieving a successful system and giving the user confidence that the new system in workable and achievable.

A. Client Connection

This is responsible for the client side communication system interface. It is used to communicate between the source and the destination. It receives the destination address, source address and the inflowing port no and binds them into a packet.

There is another sub division known as the Hopcount.This division deals with the routing of the internal message within the network. It specifies the intermediate systems to which the packet passes before going to the specified destination.

The specified intermediate systems deals with handling the messages receive from and external network and in is routing to the mentioned system. Its role is to receive the packet and identify the next intermediate system and send the packet to it. If no passers are identified the in sends the packet to the server.

B. *IDS Implementation* (Analyzing the real time network behavior)

This is the server side interface which is preset in the server system and is solely under the control of the administrator. Any transaction in the network will be monitored by the Server. It receives the packet and reads the header information from the packet such as the Destination address, Source address, Port no. It sends each andevery.Inflowing packets header information's to the chromconvert module and then receives the converted real-time Chromosomes. The real time chromosomes are checked with the rule sets. If the particular chromosomes matches with the rules provided in the rule set, It takes the decision of whether allow or block depending on which rule set it matches.

The server also monitors the intermediate systems through which the packets are passed. These passer systems are also taken into account to judge whether or not a real-time connection is considered an intrusion.

C. Chromosome Conversion

The collected attributes are converted into Chromosomes within the range and in the same behavior. The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the problem to be solved.

According to the attributes of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. These positions are sometimes referred to as genes and are changed randomly within a range during evolution. The set of chromosomes during a stage of evolution are called a population. An evaluation *function* is used to calculate the "goodness" of each chromosome

For example a real time behavior rule looks like

If {the connection has following information: source IP address 124.12.5.18; destination IP address: 130.18.206.55; destination port number: 21; connection time: 10.1 seconds}

Then {stop the connection}

The following rule is converted into Chromosomes as:

(d, 1, 0, b, -1, -1, -1, -1, 8, 2, 1, 2, b, -1, -1, -1, 4, 2, 3, 3,
5, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 4, 8, 2, 1, 1, 2, 0, 0, 0,
0, 0, 0, 7, 3, 2, 0, 0, 0, 0, 0, 0, 3, 8, 8, 9, 1).

D. Genetic Algorithm Implementation

The Genetic Algorithm is implemented, for selecting the best rule for matching with the connection. During evaluation, the selection of chromosomes for survival and combination is biased towards the fittest chromosomes. The members of this initial population are each evaluated for their *fitness* or goodness in solving the problem.

From the initial population of chromosomes, a new population is generated using three genetic operators: *reproduction*, *crossover*, and *mutation*.

- These are modeled on their biological counterparts.
- With probabilities proportional to their fitness, members of the population are selected for the new population.
- Pairs of chromosomes in the new population are chosen at random to exchange genetic material, their bits, in a mating operation called crossover. This produces two new chromosomes that replace the parents.
- Anomaly chosen bits in the offspring are flipped, called mutation.

The new population generated with these operators replaces the old population.

- The algorithm has performed one generation and then repeats for some specified number of additional generations.
- The population evolves, containing more and more highly fit chromosomes.
- When the convergence criterion is reached, such as no significant further increase in the average fitness of the population, the best chromosome produced is decoded into the search space point it represents.

VI. FUTURE EXTENSIONS

Future work includes creating a standard test data set for the genetic algorithm proposed and applying it to a test environment. Detailed specification of parameters to consider for genetic algorithm should be determined during the experiments. Combining knowledge from different security sensors into a standard rule base is another future.

VII. CONCLUSION

The software development is very flexible and much functionality can be added to it, to enhance performance of this paper titled "Intrusion Detection System In networking Using Genetic Algorithm".

By using genetic algorithm, during run time the new set of rules will added in the dataset. A brief overview of Intrusion Detection System, Genetic algorithm, and related detection techniques are discussed. This implementation of genetic algorithm is unique as it considers both temporal and spatial information of network connections during the encoding of the problem; therefore, it should be more helpful for identification of network anomalous behaviors.

REFERENCES

[1] *Network Intrusion Detection*: Evasion, Traffic Normalization and End-to-End Protocol Semantics. Mark Handley and Vern Paxson.

[2] *Intrusion Detection Based On Hidden Markov Model*. Qing-Bo Yin, Li-Ran Shen, Ru-Bo Zhang, Xue-Yao Li, Hui-Qiang Wang.

[3*] A Hidden Markov Models-Based Anomaly Intrusion Detection Method*. Ye Du, Huiqiang Wang and Yonggang Pang.

[4] *What is Computer Security*. Matt Bishop

[5] Protocol Scrubbing: *Network Security through Transparent Flow Modification*. David Watson, Matthew Smart, G. Robert Malan, Farnam Jahanian.

[6] *An Active Network–Based Intrusion Detection And Response Systems*. Han-Pang Huang, Chia-Ming Chang.